Dealing With Software: the Research Data Issues

Dealing With Software: The Research Data Issues

Neil Chue Hong
Software Sustainability Institute
JCMB, Mayfield Road
Edinburgh, EH9 3JZ, UK
N.ChueHong@software.ac.uk

ABSTRACT

1. Introduction

Software underpins much of the scientific research undertaken today. As well as the "traditional" use of software for modelling and simulation, it is used to manage and control instruments, and analyse and visualise data. This permeation of the use of software into the mainstream of research across all disciplines has meant that it is increasingly difficult to reproduce and reuse the work of other researchers. The reproducible research principle requires the full computational environment to be published as well as the paper where the results are reported. This raises the question of whether the current data management policies and infrastructure provided by universities such as Edinburgh are capable of handling software as a digital research output through its lifecycle.

2. Software and Research: the Lifecycle

We can consider the software lifecycle to have broadly the following stages, which may overlap:

- Design: where the implementation of a concept or algorithm, or extension of an existing piece of software is planned
- Development: where the software is actively being revised and worked on
- Use: where the software is actively being used in the process of carrying out research
- Preservation: where the software needs to be archived for future design, development or use.

Each stage requires a different set of tools, services and infrastructure to support it efficiently. More importantly, it must be easy to "hand off" between one stage and another – something which is often tricky for both technical and administrative policy reasons.

2.1 Design

The design stage is often considered the stage that requires the least support. However it is important that the decisions made here are recorded so that as the software goes through the stages in its lifecycle, knowledge is not lost. Principal infrastructure is concerned with knowledge management and communication: for example, wikis, mailing lists and websites.

2.2 Development

Software development is supported by a well-established set of infrastructure, such as version control / source code repositories, test and development computing infrastructure. Although it is possible to run this infrastructure at an institutional, departmental or even individual level, it is increasingly the case that it is more efficient and effective to use one of the many established third-party services (e.g. GitHub[1], SourceForge[2]). This is driven especially by the need for inter-institutional collaboration and promotion.

2.3 Use

The two concerns of this stage are making the software available for others, and providing infrastructure for the software to run on. Nowadays the first is covered by the tooling and services used for the design and development stages; the important criteria is that these services must enable a smooth transition to the use stage. For instance, it should be easy to let users sign up to mailing lists from your website, it should be easy too understand how issues are recorded and resolved. The second concern we do not cover here, though increasingly this is strongly segmented between three types of infrastructure: local computing (such as personal desktops/laptops), specialist dedicated hardware (which have been purchased specifically to run research software e.g. ECDF, regional Tier 1 infrastructure, national services like ARCHER and DIRAC) and cloud-based commodity computational infrastructure.

2.4 Preservation

The final stage is perhaps the most interesting, as is concerned with where the two very separate fields of software development infrastructure and digital preservation infrastructure meet. The various purposes why one might consider preserving software [1][2] include to achieve legal compliance and accountability; to create heritage value; to enable continued access to data and services; and to encourage software reuse.

There are two separate concerns when looking at the preservation of software. Firstly, the software must be deposited somewhere where the depositor can both have reasonable assurances of persistent storage, with appropriate metadata to allow for retrieval and replay. Secondly, there must be mechanisms in place to reference this instance of the software and connect it to the other digital ephemera to which it is associated such as the papers, datasets and workflows.

For storage, publicly available providers of code repositories may not provide the necessary assurances for long-term storage. Likewise these repositories are often structured and organised to support the constant evolution and development of the code, which is not appropriate for "stored" code versions. There are some generally available filestores which would be appropriate for software storage (e.g. FigShare[3]) and some which might be repurposed to allow software storage (e.g. Dryad[4]). There are

[1] GitHub: http://www.github.com/

[2] SourceForge: http://www.sourceforge.net/

[3] FigShare: http://figshare.com/

[4] Dryad: http://datadryad.org/

many mature, open, technical approaches to persistent, guaranteed storage of data (e.g. LOCKSS[5] , dSPACE[6], EPrints[7], and Fedora[8]) however it is not appropriate for each researcher developing software to operate such a repository themselves. Institutional repositories would appear to be the correct place for researchers to deposit software as they have already put in place policies to collect other research outputs from projects, yet an analysis of the current policies of the major UK university repositories shows that most either ignore, exclude or make no specific provision for software deposit. This clearly needs to change, but a case must be made – as it was for data – of the benefits and also an analysis of the ongoing costs. Additionally, guidelines must be in place to ensure that appropriate metadata is recorded to allow reuse in the future, and that this is checked on some regular basis. A wider discussion surrounding the requirements for metadata to both enable reuse as well as incentivise deposit is contained in [3].

3. The specific challenges of software

One of the biggest challenges for dealing with software as a part of the research data management lifecycle, is that whilst it is essentially a subset of research data, there are a number of issues [4] that make software a particularly specialised form of dataset. Because software development is typically dynamic and constantly evolving the question of what and when to store changes. However many issues, such as versioning and authorship, are ameliorated in the case of the specific deposit of a piece of software in relation to a publication.

A particular challenge is that of dependencies. Typically a piece of software will have different sets of dependencies: those required to run the software, and those required to make sense of the software. When considering how to preserve a single version of a piece of software in a repository, we need to consider what information is recorded at different stages in its lifecycle [5].

Recently, work by GitHub, FigShare and the Zenodo digital repository run at CERN has enabled the source code for software being developed using the GitHub infrastructure to be easily archiving for preservation in FigShare or Zenodo[9]. In an ideal world, this ability to archive software with one click of a buttin would be possible from any major source code repository to any institutional digital repository, such as DataShare at Edinburgh.

4. Conclusions

The twin movements of open access / open science and reproducible research necessitate the ability to describe, store, retrieve and reuse software associated with publications. Work done to address similar requirements for datasets can mostly be reused for software in this special case as many of the issues pertaining to the rapid development, multiple dependencies, and assignment of identifiers are simplified. Nevertheless, even though many of the technical challenges have been solved, the policy – particularly at institutional repositories – needs to be revised to acknowledge the requirement to provide facilities for depositing software. This area is one which the Software Sustainability Institute is looking to address in collaboration with others.

[5] LOCKSS: http://www.lockss.org/

[6] DSpace: http://www.dspace.org/

[7] EPrints: http://www.eprints.org/

[8] Fedora: http://fedora-commons.org/

[9] https://guides.github.com/activities/citable-code/

5. Acknowledgements

My thanks to Matt Shreeve and Brian Matthews for their contributions to the work referenced around software preservation, to Cameron Neylon for bringing together people for the Beyond Impact workshop where several of these issues were first raised, and to my colleagues at the Software Sustainability Institute for providing the inspiration and examples that informed the various topics covered in this abstract, in particular Steve Crouch, Mike Jackson, Simon Hettrick and Aleksandra Pawlik. The Software Sustainability Institute is supported by EPSRC grant EP/H043160/1. The work on clarifying the purpose and benefits of preserving software was carried out in collaboration with Curtis and Cartwright Ltd and funded by the Jisc.

6. References

[1] N. Chue Hong, S. Crouch, S. Hettrick, T. Parkinson, and M. Shreeve, “Software Preservation Benefits Framework,” Software Sustainability Institute Technical Report, 2010. Retrieved on 5th March 2012 from: http://www.software.ac.uk/attach/SoftwarePreservationBenefitsFramework.pdf

[2] N. Chue Hong, “Digital preservation and curation: the danger of overlooking software”. In *Preserving Complex Digital Objects (ed. J. Delve, D. Anderson),* 2014. p111-124. ISBN: 978-1-85604-958-0.

[3] N. Chue Hong, “Where does it go from here? The place of software in digital repositories”. Paper presented at OR 2012 The 7th International Conference on Open Repositories, Edinburgh, United Kingdom. Retrieved from: http://hdl.handle.net/1842/5905

[4] N. Chue Hong, “Software Impact – the differences from datasets”, Beyond Impact blog, May 2011. Retrieved from: http://beyond-impact.org/?p=175

[5] N. Chue Hong, “Minimal Information for reusable scientific software”. Submitted to the 2nd Workshop on Working towards Sustainable Scientific Software: Practice and Experience. 2014. DOI: http://dx.doi.org/10.6084/m9.figshare.1112528

Please consult the above papers for full references to works by other authors in each specific topic area.